On the Dynamic Technician Routing and Scheduling Problem

Victor Pillac, Christelle Guéret, Andrés Medaglia

On the Dynamic Technician Routing and Scheduling Problem

Victor Pillac[1,2], Christelle Guéret*[,3], and Andrés L. Medaglia[2]

[1] *LUNAM Université, École des Mines de Nantes, IRCCyN UMR 6597, Nantes, France*
[2] *Universidad de Los Andes, COPA & CEIBA, Bogotá, Colombia*
[3] *LUNAM Université, LISA - IUT Angers-Cholet, Angers, France*

October 9, 2012

Technical Report 12/5/AUTO
École des Mines de Nantes
France

Abstract

The Technician Routing and Scheduling Problem (TRSP) consists in routing staff to serve requests for service, taking into account time windows, skills, tools, and spare parts. Typical applications include maintenance operations and staff routing in telecoms, public utilities, and in the health care industry. In this paper we tackle the Dynamic TRSP (D-TRSP) in which new requests appear over time. We propose a fast reoptimization approach based on a parallel Adaptive Large Neighborhood Search (pALNS) and a Multiple Plan Approach (MPA). Finally, we present computational experiments on randomly generated instances.

Keywords: Dynamic Vehicle Routing, Technician Routing and Scheduling, Parallel Adaptive Large Neighborhood Search, Multiple Plan Approach

*Corresponding author: gueret@mines-nantes.fr

1 Introduction

The Technician Routing and Scheduling Problem (TRSP) deals with a limited crew of technicians $\mathcal{K}$ that serves a set of requests $\mathcal{R}$. The TRSP can be seen as an extension of the Vehicle Routing Problem with Time Windows (VRPTW), where technicians play the role of vehicles and requests are made by clients. In the TRSP, each technician has a set of skills, tools, and spare parts, while requests require a subset of each. The problem is then to design a set of tours such that each request is visited exactly once, within its time window, by a technician with the required skills, tools, and spare parts. The TRSP naturally arises in a wide range of applications, including telecoms, public utilities, and maintenance operations.

This problem introduces compatibility constraints between technicians and requests. While skills are intrinsic attributes, technicians may carry different tools and spare parts over the planning horizon. Technicians usually start their tour from their home with an initial set of tools and spare parts that allows them to serve an initial set of requests. They also have the opportunity to replenish their tools and spare parts at a central depot at any time to serve more requests. Tools can be seen as renewable resources, while spare parts are non renewable and consumed once the technician serves a customer.

Figure 1 illustrates an instance of the TRSP with two technicians and six requests. Technician A has the green skill, while B has both green and blue skills. Technician A starts its tour at home (gray diamond) with a hammer and a screwdriver, then serves requests 1, 2, and 3, before returning home. Technician B first serves 4, then goes to the central depot (black square) to pick up a drill that allows him/her to serve request 6 after serving request 5. Note that although request 5 is close to the tour of technician A, only technician B can serve it due to skill constraints.

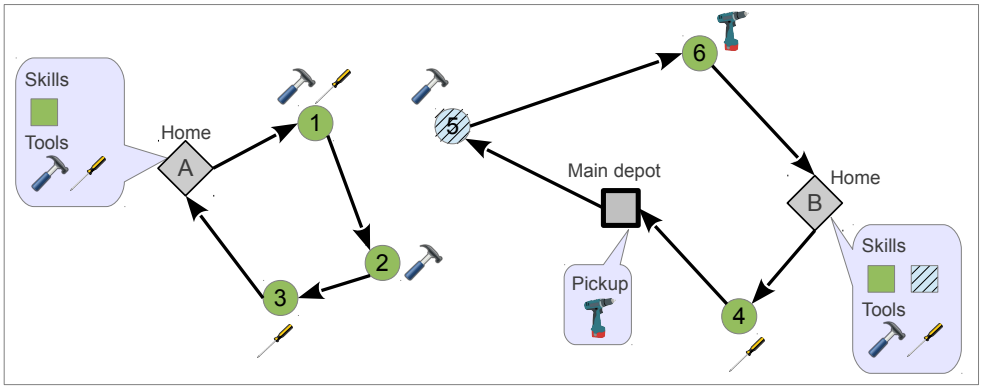

Figure 1: Example of a technician routing and scheduling problem with two technicians, three tools, and two skills

The static definition of the TRSP was introduced by the authors in Pillac et al. [22]. In this work, we tackle a dynamic variant of the problem, namely the D-TRSP, in which new requests appear while the technicians are executing their routes. In this context, two types of decisions have to be taken in real time. First, whenever a technician finishes serving a request, it must be decided what will be the next request to visit. Second, whenever a request appears, the algorithm must decide whether it is possible or desirable to accept it or not. If not the request is said to be *rejected*, it leads to a cost penalty corresponding to

the outsourcing/postponing of the request.

Despite its numerous practical applications and its challenging features, static technician routing and scheduling problems have received limited attention until recently, and to the best of our knowledge, no study simultaneously considers skills, tools, and spare parts. For instance, Xu and Chiu [37] studied the Field Technician Scheduling Problem (FTSP) seen as a variant of the VRPTW, in which the objective is to maximize the number of requests served while accounting for skill constraints, request priorities, multiple depots, and overtime. The authors describe a mixed integer formulation and develop three heuristics including a GRASP algorithm. Similarly, Weigel and Cao [36] present a software solution developed for Sears, a US retailer that serves its customers with home delivery and on-site technical assistance. The proposed solution works by first assigning technicians to requests, and then optimizing technician routes individually. Tsang and Voudouris [33] studied the technician workforce scheduling problem faced by British Telecom. Their study does not consider skill constraints, but uses a proficiency factor that reduces the service time depending on the technician experience. They propose a Fast Local Search and a Guided Local Search to solve this problem. Borenstein et al. [7] extended this problem accounting for dynamic requests and skill compatibility constraints. They cluster the requests using a k-mean algorithm followed by a heuristic that assigns technicians to areas. Finally, they propose a rule-based system that assigns and sequences the requests. They conclude their study by assessing the impact of soft clustering and show that it can increase system performance under certain assumptions.

Maintenance operations planning is a problem closely related to the TRSP. Blakeley et al. [6] present the optimization of periodic maintenance operations for Schindler Elevator Corporation in North America, a company that manufactures, installs, and maintains elevators and escalators. The problem faced by this company consists in designing a set of routes for technicians to perform periodic maintenance and repairs taking into account travel times, working regulations, and skill constraints. A similar application was studied by Tang et al. [31] who formulate the problem as a multi-period maximum collection problem in which time-dependent rewards are granted for the completion of a request. This approach allows the modeling of soft constraints such as the desirability of performing a task in a given day (*job-to-time penalties*). The authors propose a Tabu Search (TS) algorithm that yields near-optimal solutions on real instances in reasonable time.

In 2007 the French Operations Research Society (ROADEF) organized a challenge based on a problem submitted by France Telecom. The problem consists in finding a schedule for technicians to execute a set of tasks on a multiple-day horizon. Each task requires one or more skills with different minimum proficiency levels, while technicians can have multiple skills with a given proficiency. An important aspect is the creation of teams that work together during one day, combining the skills of their members, and the possibility to outsource the execution of a task. However, this problem ignores the routing aspects. Cordeau et al. [11] proposed a mathematical model and an Adaptive Large Neighborhood Search (ALNS), while Hashimoto et al. [15] proposed a Greedy Randomized Adaptive Search procedure (GRASP) approach to tackle this problem.

Work regulation is an important aspect of technician routing and scheduling. Tricoire [32] presents a technician routing problem faced by Veolia, a water distribution and treatment company. In this application, technicians have the skills to perform all requests that

are divided in two categories: user requested interventions and company scheduled visits. As new requests appear on a daily basis, the routing of technicians is performed on a rolling horizon, taking into account work regulations and customer service standards. The main contributions are a column generation approach and a memetic algorithm [8]. Their approaches take advantage of partial solutions from previous plans in the rolling horizon framework to reduce computational times.

A number of technological advances have led to a renewed interest in dynamic vehicle routing problems, leading to the development of new optimization approaches. Pillac et al. [19] classify dynamic routing problems in two categories: *deterministic* and *stochastic*. In both cases the information available to the stackholder changes over time. In stochastic setting, data is available on the dynamically revealed information in the form of known probability distributions, while in deterministic problems, changes are simply unpredictable. The present work falls in the dynamic and deterministic category, for which approaches are based either on *periodic reoptimization* or *continuous reoptimization*.

Periodic reoptimization approaches start at the beginning of the day by producing an initial set of routes that are communicated to the vehicles. As the available information is updated along the day, or at given intervals of time, an optimization is performed using the currently available information to update the routing. Such approaches can be based on algorithms developed for static problems and are therefore relatively easy to implement, however, they may introduce delays between the update of the information and the routing plan. Such approaches include the Ant Colony Systems (ACS) proposed by Montemanni et al. [18] to solve the Dynamic VRP (D-VRP). A novel feature of their approach is the use of the pheromone trace to transfer characteristics of a good solution between reoptimizations. ACS was also used by Gambardella et al. [12] and Rizzoli et al. [26]. Other heuristic approaches, such as Tabu Search (TS), were also used to tackle the Dynamic Pickup and Delivery Problem (D-PDP) [2, 9] and the Dynamic Dial-a-Ride Problem (D-DARP) [1, 3].

Continuous reoptimization approaches run throughout the day and are generally based on an adaptive memory [30] that stores alternative solutions. The adaptive memory is then used to react to changes in the available information, thus avoiding a complete reoptimization of the problem. Gendreau et al. [13] developed a parallel TS with adaptive memory to tackle a Dynamic VRPTW (D-VRPTW), that was later applied to the D-VRP [16, 17]. Bent and Van Hentenryck [4] generalized this framework and introduced the Multiple Plan Approach (MPA) to tackle the D-VRPTW. Following a different approach, Benyahia and Potvin [5] studied the Dynamic Pickup and Delivery Problem (D-PDP) and proposed a Genetic Algorithm (GA) that models the decision process of a human dispatcher. More recently, GAs were also used for the same problem [10, 14] and for the D-VRP [34].

To the best of our knowledge, no work considers simultaneously skills, tools, spare parts, and dynamically arriving requests, four important components of technician routing and scheduling. The present work addresses this aspect and proposes two optimization approaches for the dynamic version of the problem, noted D-TRSP, where new requests arrive during the execution of the routes. Section 2 introduces a fast reoptimization approach based on a parallel adaptive large neighborhood search; then Section 3 introduces a continuous reoptimization algorithm based on a multiple plan approach; finally, Section 4 presents the computational results and Section 5 concludes this paper and draws directions for future research.

2 A fast reoptimization approach

The proposed approach is based on the parallel Adaptive Large Neighborhood Search (pALNS) reoptimization algorithm introduced by Pillac et al. [21], which is used to first compute an initial solution, and then to reoptimize the solution whenever a new customer request arrives. The pALNS extends the Adaptive Large Neighborhood Search (ALNS) algorithm [23], which in turn is an extension of the Large Neighborhood Search (LNS) algorithm [24, 28]. LNS works by successively destroying (removing customers) and repairing (inserting customers back) a current solution, using *destroy* and *repair operators*. ALNS adds a layer that randomly selects operators depending on their past performance, allowing a self-adaptive version of the algorithm to the instance at hand.

Algorithm 1 presents the outline of pALNS as introduced in Pillac et al. [21]. The algorithm maintains a pool $\mathcal{P}$ of N promising solutions that are optimized in K subprocesses (note that $N \geq K$). For each *master* iteration, a subset of K promising solutions is selected randomly (line 3) and distributed among independent subprocesses. Then for I^p iterations, each subprocess selects destroy and repair operators with a roulette wheel that adaptively reflects their past performance (line 7). The resulting new solution is either accepted as the subprocess current solution or rejected according to a simulated annealing criterion (line 9), the weights of the destroy and repair operators are updated depending on their performance (line 15). The final current solution is added to the pool of promising solutions (line 17) and a filtering procedure ensures that the pool contains at most N solutions, including the best solution found so far (line 19). The algorithm stops after I^m master iterations, which corresponds to $I = I^m \times I^p$ ALNS iterations.

Algorithm 1 Parallel Adaptive Large Neighborhood Search (pALNS) algorithm

Input: $\mathcal{P}$, initial solutions; z, evaluation function; Θ^-/Θ^+, set of destroy/repair operators; N, maximum size of the solution pool; Km number of subprocesses; I^m, number of master iterations; I^p, number of iterations performed in parallel.
Output: $\mathbf{\Pi}^*$, the best solution found

1: $\mathbf{\Pi}^* \leftarrow \arg\min_{\mathbf{\Pi} \in \mathcal{P}} \{z(\mathbf{\Pi})\}$
2: **for** I^m iterations **do**
3: $\quad \mathcal{P}' \leftarrow \texttt{selectSubset}\,(\mathcal{P}, K)$ ▷ Select a subset of K solutions
4: $\quad$ **parallel forall** $\mathbf{\Pi}$ **in** $\mathcal{P}'$ **do**
5: $\quad\quad \mathbf{\Pi}^p \leftarrow \mathbf{\Pi}$ ▷ Current solution for this subprocess
6: $\quad\quad$ **for** I^p iterations **do**
7: $\quad\quad\quad d \leftarrow \texttt{select}\,(\Theta^-)\,; r \leftarrow \texttt{select}\,(\Theta^+)$ ▷ Select destroy/repair
8: $\quad\quad\quad \mathbf{\Pi}' \leftarrow \texttt{r}\,(\texttt{d}\,(\mathbf{\Pi}^p))$ ▷ Destroy and repair current solution
9: $\quad\quad\quad$ **if** $\texttt{accept}\,(\mathbf{\Pi}', \mathbf{\Pi}^p)$ **then**
10: $\quad\quad\quad\quad \mathbf{\Pi}^p \leftarrow \mathbf{\Pi}'$ ▷ $\mathbf{\Pi}'$ is accepted as current solution
11: $\quad\quad\quad$ **end if**
12: $\quad\quad\quad$ **if** $z(\mathbf{\Pi}') < z(\mathbf{\Pi}^*)$ **then**
13: $\quad\quad\quad\quad \mathbf{\Pi}^* \leftarrow \mathbf{\Pi}'$ ▷ $\mathbf{\Pi}'$ is the best solution found so far
14: $\quad\quad\quad$ **end if**
15: $\quad\quad\quad \texttt{updateScore}\,(d, r, \mathbf{\Pi}')$ ▷ Update d and r scores
16: $\quad\quad$ **end for**
17: $\quad\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{\Pi}^p\}$ ▷ Add $\mathbf{\Pi}^p$ to the pool $\mathcal{P}$
18: $\quad$ **end forall**
19: $\quad \mathcal{P} \leftarrow \texttt{retain}\,(\mathcal{P}, \mathbf{\Pi}^*, N)$ ▷ Retain at most N solutions in the pool $\mathcal{P}$
20: **end for**
21: **return** $\mathbf{\Pi}^*$

the pALNS algorithm uses three destroy operators (random, related, and critical), and three repair operators (regret-1, regret-2, regret-3). The promising solution pool $\mathcal{P}$ maintains the N best solutions according to a fitness function that considers both the cost of the solution and its diversity relative to the other solutions in the pool. The interested reader is referred to the work by Pillac et al. [21] and Pillac et al. [22] for more details on the approach.

To tackle the D-TRSP, we modified the *related destroy* operator, which attempts to remove a subset of requests that share some characteristics. We define the *relatedness* r_{ij} of requests i and j as a measure of how related two requests are (the lower the r_{ij}, the more related i and j). The procedure starts by randomly removing a *seed* request i ($\mathcal{U} = \{i\}$), then it iteratively selects a request $i \in \mathcal{U}$, and removes the most related request $j^* = \arg\min_{j \in \mathcal{R}'} \{r_{ij}\}$ from the set of unserved requests $\mathcal{R}'$. In practice the selection process is randomized and the $\lfloor y^p |\mathcal{R}'| \rfloor$-th most related request is selected, where y is a random number in $[0, 1)$ and $p \geq 1$ is a parameter that controls the level of randomness (the lower the p, the more randomness is introduced). For the D-TRSP we introduced the a-priori relatedness, which is a precalculated metric that does not depend on the actual position of requests in the tours:

$$r^s_{ij} = \left(1 + \frac{c_{ij}}{M_c}\right)^{\theta_c} \left(1 + \frac{|b_i - b_j|}{M_t}\right)^{\theta_t} \left(2 - \frac{|\mathcal{K}_i \cap \mathcal{K}_j|}{\min\{|\mathcal{K}_i|, |\mathcal{K}_j|\}}\right)^{\theta_s} \tag{1}$$

Where M_c and M_t are scaling constants, and θ_c, θ_t, and θ_s are factors that define the weight given to each metric component. The first component, measures the geographic distance between the two requests (c_{ij}). The second is the difference of due dates b_i and b_j. The third measures the number of technicians that can serve both requests, which is modeled by the intersection of the sets $\mathcal{K}_i$ and $\mathcal{K}_j$ of technicians that can serve request i and j respectively.

The second major adaptation focuses on the objective function that considers the minimization of the total working time (i.e., the sum of traveling times, service times, and waiting times). We used the concept of forward time slack introduced by Savelsbergh [27] to efficiently evaluate the minimal duration of a tour and the cost of inserting a request in a tour.

Figure 2 presents an overview of the proposed approach. The algorithm starts by producing an initial solution S_0 by using a constructive heuristic coupled with pALNS. Then each time a new customer appears, it fixes the currently executed portion of the solution, and re-runs the pALNS for a limited number of iterations, producing an updated solution S'_t. If pALNS is able to insert the new customer request, then the customer is *accepted* and S'_t becomes the new current solution, otherwise, the customer is *rejected* and S_t remains as the current solution.

3 A Multiple Plan Approach

The second proposed approach for the D-TRSP is based on the Multiple Plan Approach (MPA) introduced by Bent and Van Hentenryck [4] to tackle the D-VRPTW. MPA is a generalization of the tabu search with adaptive memory proposed by Gendreau et al. [13]. The general idea is to populate and maintain a solution pool (the routing *plans*) that are used

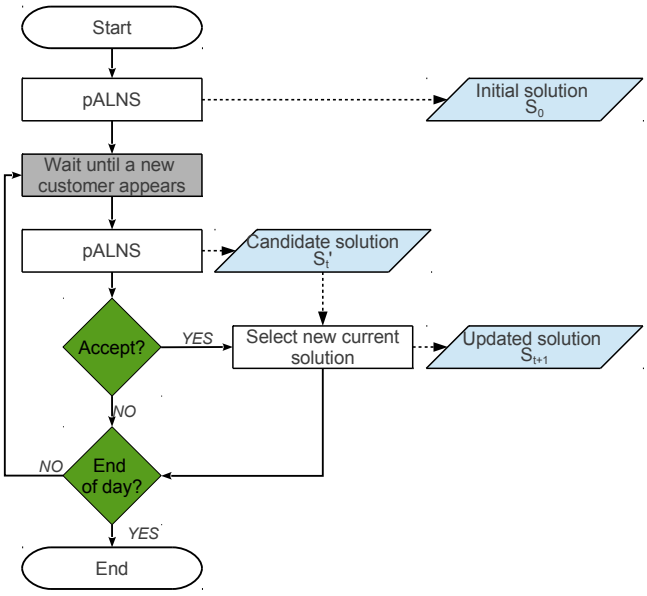

Figure 2: Overview of the proposed fast reoptimization approach

to generate a *distinguished solution*. Whenever a new request arrives, a procedure is called to check whether it can be served or not; if it can be served, then the request is inserted in each plan of the solution pool and incompatible solutions are discarded. Pool updates are performed periodically or whenever a vehicle finishes serving a customer. This pool-update phase is crucial and ensures that all solutions are coherent with the current state of vehicles and customers. The pool can be seen as an adaptive memory that maintains a set of alternative solutions.

The present work is based on the event-driven optimization framework for dynamic vehicle routing proposed by the authors, namely `jMSA` [20]. By design, `jMSA` is a flexible, parallel, and event-driven Java implementation of the Multiple Scenario Approach (MSA) [35], which is an extension of MPA for dynamic and stochastic routing problems. The proposed framework is designed to facilitate and accelerate the development and deployment of MSA-based algorithms embeddable in decision support systems.

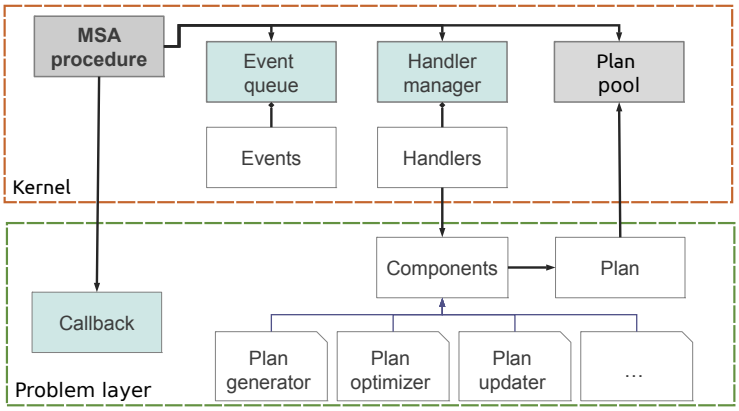

Figure 3: Design overview of the `jMSA` framework

Figure 3 outlines the main aspects of the `jMSA` framework: the *kernel* contains the com-

ponents that define the event-driven behavior, and a generic definition of the *problem layer* components. To adapt the framework for a specific problem, the user needs only to implement a subset of components, mainly to generate, optimize, and update plans. The following paragraphs give more details on how `jMSA` was adapted to tackle the D-TRSP.

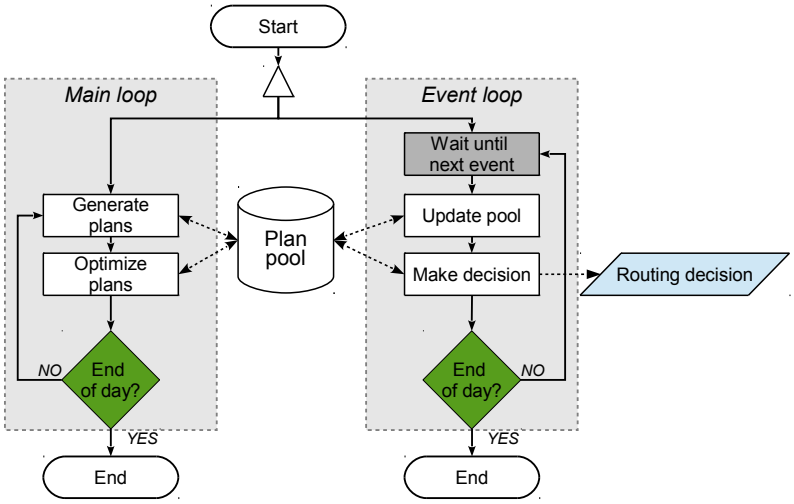

Figure 4: Overview of the multiple plan approach implemented with `jMSA`

Figure 4 presents a conceptual overview of the MPA procedure as implemented in the `jMSA` framework. `jMSA` starts two subprocesses: a *main loop* and an *event loop*. The main loop is responsible for continuously generating and optimizing a set of alternative solutions (the *routing plans*) stored in the plan pool. This main loop maximizes the utilization of the computational resources when the system is *idle*, i.e., when no decision is required. On the other hand, the event loop reacts to events from the environment, which can be of two types: a) a customer calls in and requests a service; b) a technician finishes serving a request and becomes idle. In the first case, the algorithm looks for a feasible insertion of the new request in all the solutions in the pool. If at least a given fraction of the solutions can accommodate the request, then it is accepted, otherwise it is rejected. In the second case, the algorithm selects a solution from the pool and assigns a request to all idle technicians. The event loop is also responsible for updating the pool and ensuring that all plans are coherent with the current state of the environment.

3.1 Plan generation

The goal of the plan pool is to maintain a set of diverse solutions for the current routing problem that could be used later to cope with the arrival of new requests. It is therefore necessary to have a randomized constructive heuristic that will produce a set of solutions that are both diverse and of good quality.

Our implementation is based on a randomized *regret-3* heuristic [25] which iteratively inserts requests at their best position. More formally let $\mathcal{U}$ be the set of requests that are currently not visited in the solution and let Δz_i^k be the cost of insertion of request $i \in \mathcal{U}$ in its k-th best route. The *regret-q* value r_i^q associated with request i is a measure of how desirable it is to insert i in the current iteration assuming that the best insertion will no

longer be feasible in the next iteration. It is defined as:

$$r_i^q = \sum_{k=2}^{q} \left(\Delta z_i^k - \Delta z_i^1 \right) \tag{2}$$

The randomized regret-3 algorithm iteratively selects the next request to insert using a roulette wheel in which each request is given a probability p_i:

$$p_i = \frac{r_i^3}{\sum_{j \in \mathcal{U}} r_j^3} \tag{3}$$

3.2 Optimization procedure

The optimization procedure continuously optimizes the pool of solutions. The fact that a solution might go through the optimization process more than once requires an algorithm able to escape from local optima to further improve a solution. Therefore, we implemente an Adaptive Large Neighborhood Search (ALNS) similar to the pALNS presented in Section 2. Note that the choice of having a sequential optimization algorithm is motivated by the fact that `jMSA` will optimize various plans in parallel.

Algorithm 2 outlines the ALNS algorithm. ALNS starts with an initial solution $\mathbf{\Pi}$. Then for I iterations, the algorithm selects destroy and repair operators (line 4) with a roulette wheel that reflects their past performance. The destroy operator removes a subset of requests from the current solution that are then reinserted by the repair operator (line 5). The resulting new solution is accepted as current solution according to a simulated annealing criterion (line 6). At the end of each iteration, the scores of the destroy and repair operators are updated depending on the solution they generated (line 12).

Algorithm 2 Adaptive Large Neighborhood Search (ALNS) algorithm

Input: $\mathbf{\Pi}_0$ initial solution, z evaluation function, Θ^-/Θ^+ set of destroy/repair operators, I number of iterations
Output: $\mathbf{\Pi}^*$ the best solution found

1: $\mathbf{\Pi}^* \leftarrow \mathbf{\Pi}_0$ ▷ Initialize best solution
2: $\mathbf{\Pi} \leftarrow \mathbf{\Pi}_0$ ▷ Initialize current solution
3: **for** I iterations **do**
4: $d \leftarrow$ `select` (Θ^-) ; $r \leftarrow$ `select` (Θ^+) ▷ Select destroy/repair
5: $\mathbf{\Pi}' \leftarrow$ `r` (`d` $(\mathbf{\Pi})$) ▷ Generate a neighbor
6: **if** `accept` $(\mathbf{\Pi}', \mathbf{\Pi})$ **then** ▷ $\mathbf{\Pi}'$ is accepted as current solution
7: $\mathbf{\Pi} \leftarrow \mathbf{\Pi}'$ ▷ Update current solution
8: **end if**
9: **if** $z(\mathbf{\Pi}') < z(\mathbf{\Pi}^*)$ **then** ▷ An improvement has been found
10: $\mathbf{\Pi}^* \leftarrow \mathbf{\Pi}'$ ▷ Update best solution
11: **end if**
12: `updateScore` $(d, r, \mathbf{\Pi}')$ ▷ Update scores
13: **end for**
14: **return** $\mathbf{\Pi}^*$

3.3 Interactions with the decision maker

The decision maker interacts with MPA by raising events. In the context of the D-TRSP, there are two major events: the arrival of a new request and the end-of-service of a request.

Arrival of new requests Whenever a new request appears, a procedure attempts to insert it in all the plans in the pool. The procedure starts by trying to insert the request directly, if it fails, it removes a fraction of the requests and uses regret-3 to attempt to reinsert all requests. If at least a given fraction of the plans can accommodate the new request then it is accepted and the plans are updated accordingly, otherwise the request is marked as rejected.

Real-time routing decisions When a technician finishes serving a request and becomes idle, a decision needs to be taken on what will be his/her next assignment. To this end we use the *consensus* algorithm [35] which aggregates the information contained in the plans from the pool to select a distinguished solution and assign requests to idle technicians. The intuition behind consensus is to assign to each technician the requests that appear first with the highest frequency across plans. As multiple technicians are involved, the consensus algorithm selects a solution from the pool that maximizes the consensus across all technicians. Algorithm 3 presents the details of the algorithm. Consensus starts by counting the number of times each request appears first in a tour across all solutions from the pool (lines 1 to 6). Then the algorithm evaluates each solution by summing the evaluations of the first request of each of its tours (line 11). Finally, the solution $\mathbf{\Pi}^*$ with the highest evaluation is returned, and the first unserved request of each tour in $\mathbf{\Pi}^*$ is the next assignment of the corresponding technician.

Algorithm 3 The consensus algorithm

Input: $\mathcal{P}$ a pool of alternative solutions
Output: $\mathbf{\Pi}^*$ a distinguished solution

1: $e \leftarrow [0]_{i \in \mathcal{R}}$ ▷ Initialize the evaluation of all requests
2: **for all** $\mathbf{\Pi} \in \mathcal{P}$ **do** ▷ For each solution in the pool
3: **for all** $\pi \in \mathbf{\Pi}$ **do** ▷ For each tour in the solution
4: $e[\pi_0] \leftarrow e[\pi_0] + 1$ ▷ Increment the evaluation of the first unserved request π_0
5: **end for**
6: **end for**
7: $s^* \leftarrow 0$
8: **for all** $\mathbf{\Pi} \in \mathcal{P}$ **do**
9: $s \leftarrow 0$ ▷ Initialize the evaluation of this scenario
10: **for all** $\pi \in \mathbf{\Pi}$ **do**
11: $s \leftarrow s + e[\pi_0]$ ▷ Update the evaluation of this scenario
12: **end for**
13: **if** $s > s^*$ **then**
14: $\mathbf{\Pi}^* \leftarrow \mathbf{\Pi}$
15: **end if**
16: **end for**
17: **return** $\mathbf{\Pi}^*$

Waiting strategy It is important to note that the immediate commitment of idle technicians to requests may lead to difficulties when new requests appear. Figure 5 illustrates this with a single technician. Suppose that at time t a technician is assigned to a request i, if the technician is committed immediately to i, it will travel to i then wait at its destination until the start of the time window (black brackets). On the other hand, if a waiting strategy is used, the technician will remain idle until the latest moment such that it will not wait at i.

Assume now that at time $t+1$ a new request j appears, in the first case j cannot be served as the technician is already waiting at i (the hashed section is already executed), while in the second case it can be inserted right before i.

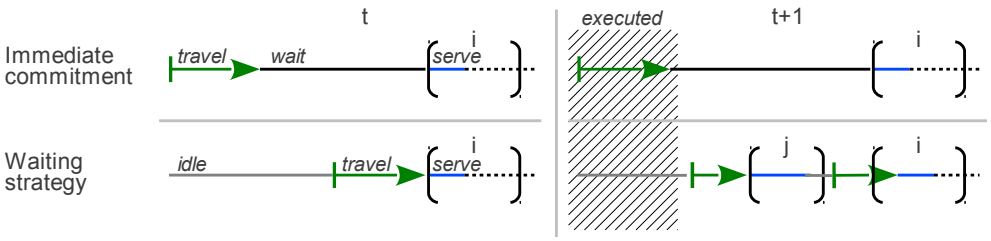

Figure 5: Advantage of a waiting strategy.

The proposed waiting strategy is implemented as follows: first, the procedure evaluates the latest departure time so that the technician will not have to wait at its next request. If this departure time is within a given range, then it is assumed there is not enough time to change the technician's route and the technician is committed to the next request. Otherwise, the technician remains in an idle state for a given time, after what a new decision is taken, leaving time for further changes in assignments.

4 Preliminary results

We tested the pALNS and MPA approaches on a set of 56 randomly generated instances based on the Solomon [29] testbed. The instances contain 100 requests located randomly (R), in clusters (C), or combining both (RC); while the planning horizon is either short (type 1) or long (type 2). These instances are organized combining location and horizon (i.e., C1, C2, R1, R2, RC1, and RC2). We considered 5 skills, 5 tools, and 5 types of spare parts. For each request, we selected 1 skill, and between 0 and 2 tools and spare part types. Each of the 25 technicians has between 2 and 4 skills, and an initial set of 0 to 5 tools, and 2 to 5 spare parts. In addition, we generated release dates for either 10, 30, 50, 70, or 90 requests, leading to a complete testbed of 280 dynamic instances.

We compare the two proposed approaches with a regret-3 heuristic. This simple approach starts with the same initial solution as pALNS. Each time a new request appears, it attempts to insert it in the current solution using a regret heuristic, rejecting it if it cannot be inserted. The parameter setting for the pALNS reoptimization approach is identical to the one presented in Pillac et al. [21], we allowed for 25,000 iterations for the calculation of the initial solution, and 5,000 for subsequent reoptimizations. The maximum pool size for MPA was set to 50 plans, while the ALNS algorithms used the same parameters as in Pillac et al. [21], with a maximum of 5,000 iterations per optimization.

The same simulation procedure was used to test the three approaches. First, the simulator allows them time to either design an initial solution (pALNS and regret-3) or initialize the pool of plans (MPA). Then the procedure simulates the routing of the technicians using an average traveling speed and taking into account waiting times. Whenever a technician becomes idle, the simulator uses the current solution (pALNS and regret-3) or distinguished plan (MPA) to select the next assignment for this technician. The simulator handles the new requests and depending on the approach response it marks them as accepted or rejected.

Finally, it finds an a-posteriori solution to the problem defined by all the accepted requests using pALNS with 50,000 iterations.

4.1 Minimizing the total working time

The static TRSP problem [22] considers the minimization of the total working time. In a dynamic setting, this objective leads to the premature ending of tours: technicians are sent home as soon as possible to reduce the duration of their tour, ignoring the fact that additional requests may appear in the future. To prevent this behavior we define a *cutoff policy* that ensures for an instance I that technicians will no be sent back to their home until time $t_c(I)$. Considering that each instance have a different horizon $[0, T(I)]$, we define the relative cutoff $\alpha(G)$ for instance group G. The value of $\alpha(G)$ is defined such that all requests of instance $I \in G$ will be known before $\alpha(G)T(I)$ with a certain probability. In our experiments, $\alpha(G)$ corresponds to the 90-percentile of the distribution of $\left\{ \frac{rd^I_{max}}{T(I)} \right\}_{I \in G}$ where rd^I_{max} is the last release date for instance I[1].

A direct consequence of this policy is that the minimal tour duration for instance I is either 0 (if the technician is not used), or $\alpha(G)T(I)$. Therefore the total duration at the end of the day is significantly longer than the one found when solving the static problem.

	pALNS		**MPA**		**regret-3**	
δ	**Gap (%)**	**R**	**Gap (%)**	**R**	**Gap (%)**	**R**
10	65.7	0.0	152.8	1.5	59.9	0.4
30	79.5	0.1	160.1	3.2	84.6	0.6
50	93.0	0.1	150.6	4.6	100.4	1.0
70	100.3	0.2	153.9	6.3	113.8	1.4
90	102.8	0.4	154.0	6.0	122.3	1.8
Avg.	**88.3**	**0.2**	**154.3**	**4.4**	**96.2**	**1.0**

Table 1: Average gap to a-posteriori solution and number of rejected requests for the D-TRSP instances minimizing the total duration.

Table 1 reports the results for the 56 instances and 5 degrees of dynamism when minimizing the total duration. The first column contains the degree of dynamism (δ) defined as the number of dynamic requests. The second and third columns report the average gap to an a-posteriori solution[2] and the average number of rejected requests (R) for the pALNS. The fourth and fifth columns contain these statistics for the MPA, and the seventh and eighth columns for the regret-3 heuristic. Note that running times for pALNS are of 12s on average for the calculation of the initial solution and 2.8s for subsequent reoptimizations, while decision times are negligible for both MPA and regret-3.

Firstly, it can be observed that gaps are large regardless of the approach. This is due to the fact that the a-posteriori solution does not consider the cutoff strategy enforced in the dynamic context. Therefore the gap should not be interpreted as an absolute performance metric, but instead as a metric that allows comparisons between approaches. Secondly, the

[1] With this definition: $\alpha^{C1} = 0.380$, $\alpha^{C2} = 0.509$, $\alpha^{R1} = 0.357$, $\alpha^{R2} = 0.419$, $\alpha^{RC1} = 0.321$, $\alpha^{RC2} = 0.400$

[2] The gap for instance I is defined as the ratio $\frac{z(I) - \underline{z}(I)}{\underline{z}(I)}$ where $z(I)$ is the value of the solution found by the algorithm for the dynamic instance, and $\underline{z}(I)$ is a lower bound obtained by solving the static a-posteriori instance with 50,000 iterations of the pALNS algorithm.

results show that, as expected, both the gap and number of rejected requests increase with the degree of dynamism. Finally, they indicate that the pALNS approach leads to better solutions both in terms of quality of the routing (measured by the gap) and ability to cope with new requests (measured by R). In contrast, MPA performs poorly and is dominated by the simpler regret-3 reoptimization approach. This can be explained by the fact that the decision process in MPA does not take into account the cost (total duration) of plans to select the distinguished plan, while the other two approaches explicitly focus on the cost. In addition, our experiments show that MPA tends to use more technicians, starting more tours than pALNS and regret-3. Considering that technicians then have to wait until the cutoff time, this leads to a greater total duration. On the other hand, the higher number of rejected requests can be explained by the fact that MPA is more conservative than the other approaches, as it requires that a fraction of the plans can accommodate the new requests, while the other approaches only require a feasible insertion in the current solution.

4.2 Minimizing the total distance

The cutoff policy forces technicians to wait at their current location before returning home. Thus, the minimization of the working time may not be as relevant in a dynamic context as it is for the static case. To assess the validity of this objective, we performed the same experiments with the minimization of the traveled distance.

	pALNS		**MPA**		**regret-3**	
δ	**Gap (%)**	**R**	**Gap (%)**	**R**	**Gap (%)**	**R**
10	2.4	0.1	9.1	1.9	10.5	0.3
30	5.4	0.1	11.0	4.6	30.5	0.4
50	10.8	0.3	14.4	5.6	44.1	1.0
70	11.8	0.2	21.3	8.7	57.5	1.2
90	17.9	0.4	23.9	8.1	64.1	1.4
Avg.	**9.7**	**0.2**	**16.1**	**5.9**	**41.3**	**0.8**

Table 2: Average gap to a-posteriori solution and number of rejected requests for the D-TRSP instances minimizing the total distance.

Table 2 compares the different approaches when the objective only considers the minimization of the traveled distance. As before, the gap and number of rejected requests generally increases with the degree of dynamism. These results show that pALNS consistently outperforms the two other approaches, both in terms of gap and number of rejected requests. However, in this case MPA is the second best-performing approach in terms of gap, but it remains third with respect to the number of rejected requests. As before, our experiments show that MPA uses more technicians on average. However, what was a disadvantage when minimizing the total duration helps MPA in reducing the total distance. Nonetheless, the remark regarding the number of rejected requests remains valid: the approach seems to be overly conservative.

Finally, Table 3 presents the effect of the change in the objective function in both total working time (Δ_{WT}) and traveled distance (Δ_{Dist}) for the three approaches. As expected, minimizing the distance instead of the working time leads to a reduction of the total traveled distance by 45%, 55%, and 32% for pALNS, MPA, and regret-3, respectively. More

	pALNS		MPA		regret-3	
δ	Δ_{WT}	Δ_{Dist}	Δ_{WT}	Δ_{Dist}	Δ_{WT}	Δ_{Dist}
10	-8.0	-40.9	-13.3	-58.1	-1.4	-33.6
30	-9.8	-45.5	-15.8	-57.2	-7.7	-31.7
50	-16.4	-46.5	-11.4	-55.1	-11.0	-31.4
70	-18.5	-47.6	-12.8	-54.1	-10.8	-30.2
90	-20.2	-45.4	-10.1	-52.9	-11.9	-32.0
Avg.	**-14.6**	**-45.2**	**-12.6**	**-55.4**	**-8.5**	**-31.8**

Table 3: Difference in total working time and distance when minimizing the total distance instead of the total working time (in %).

surprisingly, it also leads to a reduction of the total working time by 15%, 13%, and 8%. This can be explained by the cutoff policy that is contradictory with the minimization of the working, which mainly focuses on minimizing waiting times. In contrast, focusing on the minimization of the traveled distance always leads to a reduction of the travel time, which in turn reduces the duration of tours.

5 Conclusions

In this paper we introduced a new dynamic optimization problem, namely the Dynamic Technician Routing and Scheduling Problem (D-TRSP). This problem arises in numerous practical contexts such as public utilities, telecoms, and maintenance operations.

We propose two solution methods to tackle the D-TRSP. The first is a periodic reoptimization approach based on a parallel Adaptive Large Neighborhood Search (pALNS) that produces a new routing plan each time a new request appears. The second is a continuous reoptimization approach based on the Multiple Plan Approach (MPA) that continuously optimizes a pool of routing plans that are then use to take routing decisions.

Our preliminary computational results indicate that the pALNS based reoptimization approach dominates MPA and a simpler regret-3 heuristic, by yielding high quality results in limited time. In addition, its relative simplicity makes it a good candidate for practical applications. MPA results were disappointing, but this can be attributed to the decision process which does not takes into account the plan costs, and an overly conservative request acceptance criterion.

In addition, we have demonstrated that the minimization of the total working time, although perfectly sound in a static context, does not fit well in a dynamic environment. In particular, we have shown that minimizing the total distance ultimately leads to solutions that are better both in terms of total distance and duration.

Further work will focus on improving MPA to take better decisions and reject less requests. In addition, we are testing the proposed approach on real world data from an industrial partner. Finally, the uncertainty should be modeled to better anticipate the arrival of new requests and improve the quality of the decisions.

Acknowledgements Financial support for this work was provided by the CPER Vallée du Libre (Contrat de Projet Etat Region, France); and the CEIBA (Centro de Estudios Interdisciplinarios Básicos y Aplicados en Complejidad, Colombia). This support is gratefully

acknowledged.

References

[1] Attanasio, A., Cordeau, J. F., Ghiani, G., and Laporte, G. (2004). Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, doi:10.1016/j.parco.2003.12.001.

[2] Barcelo, J., Grzybowska, H., and Pardo, S. (2007). Vehicle routing and scheduling models, simulation and city logistics. In Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I., editors, *Dynamic Fleet Management*, volume 38 of *Operations Research/Computer Science Interfaces*, pages 163–195. Springer US.

[3] Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR Spectrum*, 32:77–107, doi:10.1007/s00291-008-0135-6.

[4] Bent, R. and Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

[5] Benyahia, I. and Potvin, J. Y. (1998). Decision support for vehicle dispatching using genetic programming. *IEEE Transactions on Systems Man and Cybernetics Part A - Systems and Humans*, 28(3):306–314.

[6] Blakeley, F., Arguello, B., Cao, B., Hall, W., and Knolmajer, J. (2003). Optimizing periodic maintenance operations for schindler elevator corporation. *INTERFACES*, 33(1):67–79, doi:10.1287/inte.33.1.67.12722.

[7] Borenstein, Y., Shah, N., Tsang, E., Dorne, R., Alsheddy, A., and Voudouris, C. (2010). On the partitioning of dynamic workforce scheduling problems. *Journal of Scheduling*, 13(4):411–425, doi:10.1007/s10951-009-0152-6.

[8] Bostel, N., Dejax, P., Guez, P., and Tricoire, F. (2008). Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In Sharda, R., Golden, B., Raghavan, S., and Wasil, E., editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 503–525. Springer US.

[9] Chang, M. S., Chen, S., and Hsueh, C. (2003). Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. *Journal of the Eastern Asia Society for Transportation Studies*, 5:2273–2286.

[10] Cheung, B. K. S., Choy, K. L., Li, C.-L., Shi, W., and Tang, J. (2008). Dynamic routing model and solution methods for fleet management with mobile technologies. *International Journal of Production Economics*, 113(2):694–705, doi:10.1016/j.ijpe.2007.10.018.

[11] Cordeau, J.-F., Laporte, G., Pasin, F., and Ropke, S. (2010). Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, doi:10.1007/s10951-010-0188-7.

[12] Gambardella, L., Rizzoli, A., Oliverio, F., Casagrande, N., Donati, A., Montemanni, R., and Lucibello, E. (2003). Ant colony optimization for vehicle routing in advanced logistics systems. In *Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003)*, pages 3–9.

[13] Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, doi:10.1287/trsc.33.4.381.

[14] Haghani, A. and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, 32(11):2959 – 2986, doi:10.1016/j.cor.2004.04.013.

[15] Hashimoto, H., Boussier, S., Vasquez, M., and Wilbaut, C. (2011). A GRASP-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183:143–161, doi:10.1007/s10479-009-0545-0.

[16] Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438, doi:10.1287/trsc.34.4.426.12325.

[17] Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379 – 396, doi:10.1016/S0377-2217(02)00147-9.

[18] Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343, doi:10.1007/s10878-005-4922-6.

[19] Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2011a). A review of dynamic vehicle routing problems. Technical Report CIRRELT-2011-62, CIRRELT.

[20] Pillac, V., Guéret, C., and Medaglia, A. L. (2011b). An event-driven optimization framework for dynamic vehicle routing. Technical Report 11/2/AUTO, École des Mines de Nantes, France.

[21] Pillac, V., Guéret, C., and Medaglia, A. L. (2012a). A fast re-optimization approach for dynamic vehicle routing. Technical Report 12/X/AUTO, École des Mines de Nantes, France.

[22] Pillac, V., Guéret, C., and Medaglia, A. L. (2012b). A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, Accepted manuscript.

[23] Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, doi:10.1016/j.cor.2005.09.012.

[24] Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US.

[25] Potvin, J.-Y. and Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, doi:10.1016/0377-2217(93)90221-8.

[26] Rizzoli, A., Montemanni, R., Lucibello, E., and Gambardella, L. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(sa):135–151.

[27] Savelsbergh, M. (1992). The vehicle routing problem with time windows: minimizing route duration. *INFORMS*, 4(2):146–154, doi:10.1287/ijoc.4.2.146.

[28] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg.

[29] Solomon, M. M. (1987). Algorithms for the vehicle-routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265.

[30] Taillard, E. D., Gambardella, L. M., Gendreau, M., and Potvin, J.-Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135(1):1 – 16, doi:10.1016/S0377-2217(00)00268-X.

[31] Tang, H., Miller-Hooks, E., and Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E: Logistics and Transportation Review*, 43(5):591–609, doi:10.1016/j.tre.2006.03.004.

[32] Tricoire, F. (2006). *Optimisation des Tournées de Véhicules et de Personnels de Maintenance : Application à la Distribution et au Traitement des Eaux*. PhD thesis, École Nationale Supérieure des Techniques Industrielles et des Mines de Nantes. EDSTIM 366-250.

[33] Tsang, E. and Voudouris, C. (1997). Fast local search and guided local search and their application to british telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119 – 127, doi:10.1016/S0167-6377(96)00042-9.

[34] Van Hemert, J. I. and Poutré, J. L. (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. In Yao, X., Burke, E., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J., Tino, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 692–701. Springer Berlin / Heidelberg.

[35] Van Hentenryck, P. and Bent, R. (2006). *Online stochastic combinatorial optimization*. MIT Press.

[36] Weigel, D. and Cao, B. (1999). Applying gis and or techniques to solve sears technician-dispatching and home delivery problems. *INTERFACES*, 29(1):112–130, doi:10.1287/inte.29.1.112.

[37] Xu, J. and Chiu, S. (2001). Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509.